НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 “Комп’ютерні науки ”

на тему “Створення БД для аналізу території з метою розміщення вітроенергетичних станцій”

Виконав: студент 4 курсу, групи ТМ-52

Литка Сергій Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент Ковальчук А. М. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту

Литці Сергію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Створення БД для аналізу території з метою розміщення вітроенергетичних станцій”

керівник роботи Ковальчук Артем Михайлович доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”22”05 2019р. № 1325С

2. Строк подання студентом роботи 17.06.2019-21.06.2019

3. Вихідні дані до роботи платформа Microsoft Visual Studio 2015, мова програмування C# з технологією .NET Framework 4.5, програмний продукт ArcGIS та технологія Windows Forms для створення інтерфейсу користувача.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Створити базу даних метеорологічних умов. Створити базу даних вітроенергетичних установок та їх характеристик. Розрахувати обсяги генерування енергії за визначений проміжок часу. Створити зручний користувацький інтерфейс для вводу початкових даних та отримання результату у зручному форматі. Візуально зобразити результати у вигляді графіків.

5. Перелік ілюстративного матеріалу
«Мета роботи», «Постановка задачі», «Аналіз існуючих програмних засобів», «Засоби розробки», «Структура системи», «Алгоритм роботи програми», «Приклад роботи програми», «Результати роботи програми», «Висновки».

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "14" жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	1 Затвердження теми роботи	09.10.2018	
2.	Вивчення та аналіз задачі	14.10.2018-23.12.2018	
3.	Розробка архітектури та загальної структури системи	02.02.2019-03.03.2019	
4.	Розробка структур окремих підсистем	04.03.2019-14.04.2019	
5.	Програмна реалізація системи	15.04.2019-19.05.2019	
6.	Оформлення пояснювальної записки	20.05.2019-05.06.2019	
7.	2 Захист програмного продукту	24.05.2019	
8.	3 Передзахист	31.05.2019	
9.	Захист	17.06.2019-21.06.2019	

Студент _____ Литка С. С.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Ковальчук А. М.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

М

е

т

о

ю

характеристик вітроенергетичної установки. Користувацький інтерфейс програми дозволяє ввести необхідні вхідні дані, серед яких вибір території на якій буде проводитись розрахунок, вибір періоду та інші. При обробці даних додаток виводить графіки розподілу вітрового потенціалу, криву потужності вітроенергетичної установки та обчислює сумарні обсяги енергії згенерованої за період дослідження.

Записка містить 50 сторінок, 16 рисунків, 6 таблиць та 22 посилання.

ABSTRACT

The purpose of the work was to create a software product that allows you to calculate the amount of energy generated over a certain period of time based on meteorological conditions (wind activity) and bases of power characteristics of the wind power plant. The user interface of the program allows the user to enter the necessary input data, including the choice of territory for which the calculation will be carried out, range selection and others. While processing the data, the application displays the chars of wind power distribution, wind power curve and calculates the total amount of energy generated during the study period.

The note contains 50 pages, 16 images, 6 tables and 22 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CLR — “загальномовне виконуюче середовище” — це компонент пакету Microsoft .NET Framework, віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

АЕС — атомна електростанція

БД — база даних

ВЕС — вітроенергетична станція

ВЕУ — вітроенергетична установка

ОС — операційна система

СУБД — система управління базами даних

ТЕС — теплоенергетична станція

ВСТУП

Протягом року на нашу планету надходить в 15 тис. разів більше енергії від обсягів нинішнього споживання всіма країнами світу. На енергію вітру перетворюється близько 3% енергії сонячного випромінювання, тому ресурси енергії вітру на Землі приблизно у 50 разів більші за сумарні енергетичні потреби всього людства.

Енергію вітру людина використовує із незапам'ятних часів. Спочатку це був парус, потім вітровий млин. Сучасні вітряки, що виробляють електроенергію, з'явились лише в XX столітті.

Територія нашої країни має значні ресурси вітрової енергії, які оцінюються у 30 ТВт × год/рік за даними Міжгалузевого науково-технічного центру вітроенергетики Національної академії наук України.

На території України придатними для будівництва ВЕУ вважаються площі до 7 тис. км^2, це — донецький, карпатський, гірнокримський, західнокримський, керчинський, приазовський регіони, Полтавська та Харківська області. За розрахунками науковців, можна було б одержувати електроенергію в обсягах, які б надавали можливість забезпечити до 50% загального енергоспоживання країни, при максимальному використанні сили вітру в цих регіонах.

Метою роботи є розробка програмного продукту для аналізу території України для визначення ефективності впровадження вітроенергетичної установки та розрахунку обсягів генерування енергії за вказаний проміжок часу.

Для розробки програмного забезпечення було використано середовище розробки Microsoft Visual Studio 2015, мову програмування C#, програмну технологію .NET Framework 4.5, програмний продукт ArcGIS, а також технологію Windows Forms для створення інтерфейсу користувача. Системою керування базами даних було обрано MS SQL Server 2017. Також для розробки було використано операційну систему Windows 8.1.

Перший розділ пояснювальної записки містить постановку задачі та призначення програмного забезпечення; в другому розділі надано опис предметної області та існуючих програмних засобів; третій розділ містить опис засобів реалізації програмного продукту; в четвертому розділі надано опис програмної реалізації; у п’ятому розділі описана методика роботи користувача з системою.

4 ПОСТАНОВКА ЗАДАЧІ

Метою бакалаврської роботи є створення програмного продукту, що дозволяє розрахувати обсяги генерування енергії за визначений проміжок часу на основі метеорологічних умов (вітрова активність) та бази енергетичних характеристик ВЕУ.

Оцінювання ефективності впровадження електростанцій традиційного типу (теплових і атомних) випливає з двох основних чинників:

— наявності і стабільності інформації щодо енергетичних параметрів енергоносіїв;

— повільності технічного прогресу щодо обладнання ТЕС і АЕС.

Разом з тим, специфіка вітрових електростанцій характеризується протилежними факторами:

— непередбачуваним характером вітру як енергоресурсу;

— відсутністю відповідної інформації щодо енергетичних параметрів вітру;

— бурхливим технічними розвитком в конструюванні і виробництві вітрових електроустановок.

Все це, взяте разом, значно ускладнює процес оцінки ефективності ВЕУ і обґрунтування доцільності їх використання. Перші методичні матеріали в даному напрямку було розроблено в 80-х роках XX ст. в Данії і США — країнах-піонерах вітроенергетики. Рівень глибини і точності обґрунтувань ефективності проектів ВЕУ на той час відповідав невеликим потужностям ВЕУ, будівництво яких повністю або значною мірою здійснювалось за державні кошти. В 90-х роках XX ст. з початком широкомасштабного розвитку світової вітроенергетики, з набуттям нею “бізнесопридатності”, в даній галузі енергетики почали реалізовуватись інвестиційні проекти потужних ВЕС. Такі проекти є капіталомісткими і потребують залучення значних коштів, в тому числі з ринку капіталу. Комерційні ж банки і інвестиційні фонди, що здійснюють фінансування інвестиційних проектів ВЕС, вкладаючи значні кошти, вимагають у інвесторів-позичальників доказів реальності відповідних проектів.

Постановка задачі бакалаврської роботи виглядає наступним чином:

1. Створити базу даних метеорологічних умов.

2. Створити базу даних вітроенергетичних установок та їх характеристик.

3. Забезпечити користувача інструментами:

— вибору території для оцінки;

— вводу періоду на протязі якого буде здійснюватись оцінка.

4. Розрахувати обсяги генерування енергії за визначений проміжок часу.

5 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ВІТРОЕНЕРГЕТИЧНОЇ УСТАНОВКИ

Існуючі системи для розрахунку обсягів генерування енергії є зразками програмного забезпечення, що створені для візуалізації та моніторингу результатів роботи певних типів вітроенергетичних установок. Розглянемо деякі приклади таких програмних систем.

2.1 Програмне забезпечення Wind Power

Система Wind Power — це програмне забезпечення розроблене для персональних комп'ютерів, яке може бути використано для розрахунку продуктивності вітроенергетиних установок, починаючи від невеликих вітчизняних установок з роторами діаметром до двох метрів до найбільших комерційних одиниць з діаметрами ротора близько 100 метрів в діаметрі [1]. Цей додаток може бути використаний як для вітрових турбін з горизонтальною віссю (HAWTs) [2], так і для вітрових турбін з вертикальною віссю (VAWT) [3]. Він обчислює середню потужність та річне вироблення енергі, а також дозволяє оцінювати економічність конкретної установки.

Основні можливості програми Wind Power (рисунок 2.1):

— обчислення середньої потужність і річного виходу енергії;

— оцінка рентабельності інвестицій у ВЕУ;

— оцінка періоду окупності ВЕУ;

— оцінки вартості за кіловат-годину;

— профіль вихідної потужності, що показує відсоток часу, за який ВЕУ виробляє різні потужності, включаючи нульову потужність;

— для установки, що використовує місцеву енергію, можна оцінити, скільки енергії буде використано на місцевому рівні та скільки буде експортовано до мережі.

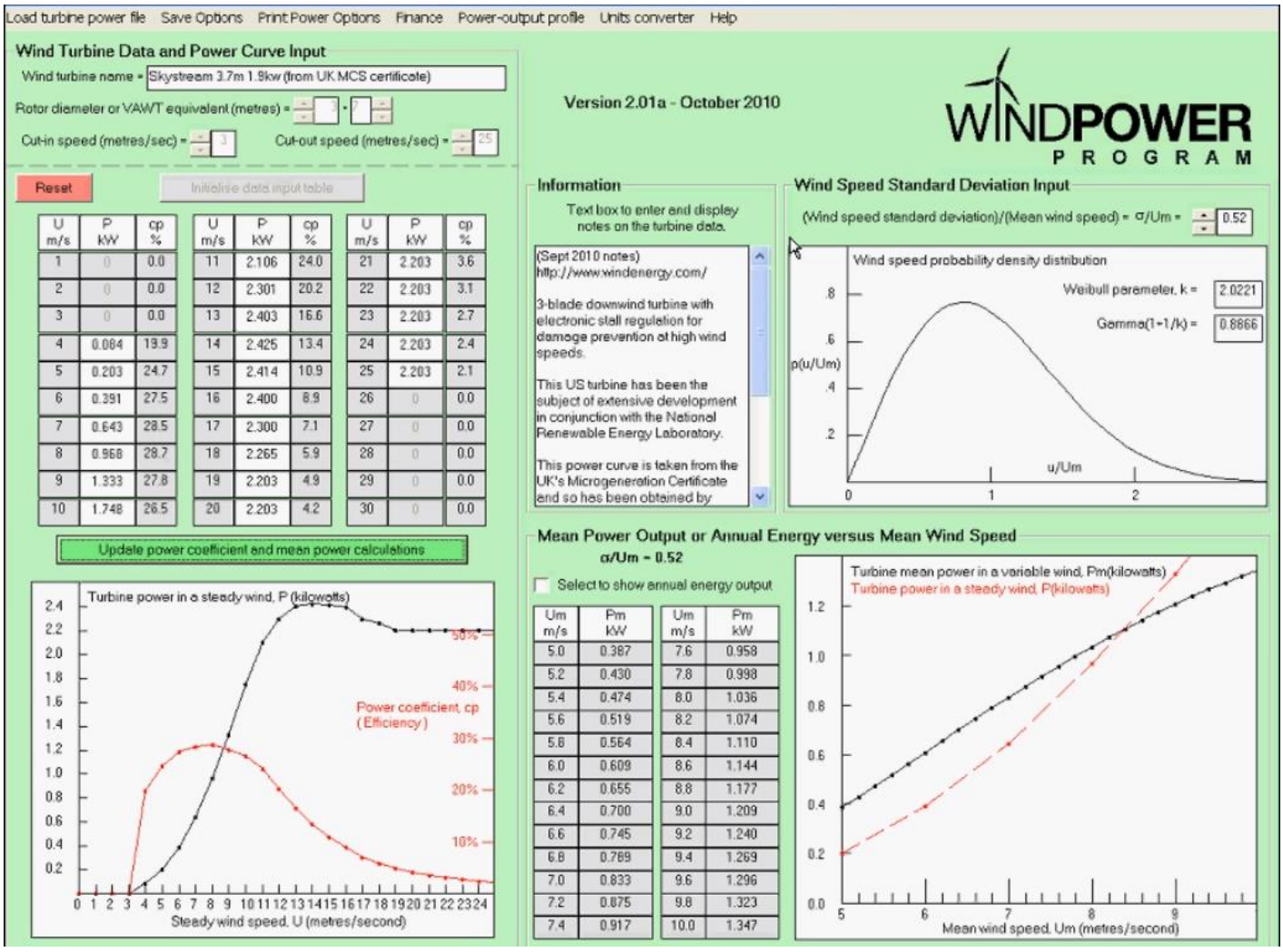


Рисунок 1.1 — Загальний вигляд програми Wind Power

Області застосування програмного забезпечення Wind Power:

— звичайний користувач зацікавлений в маленькій ВЕУ, який хоче, оцінити продуктивність і економічність її можливого встановлення (рисунок 2.2);

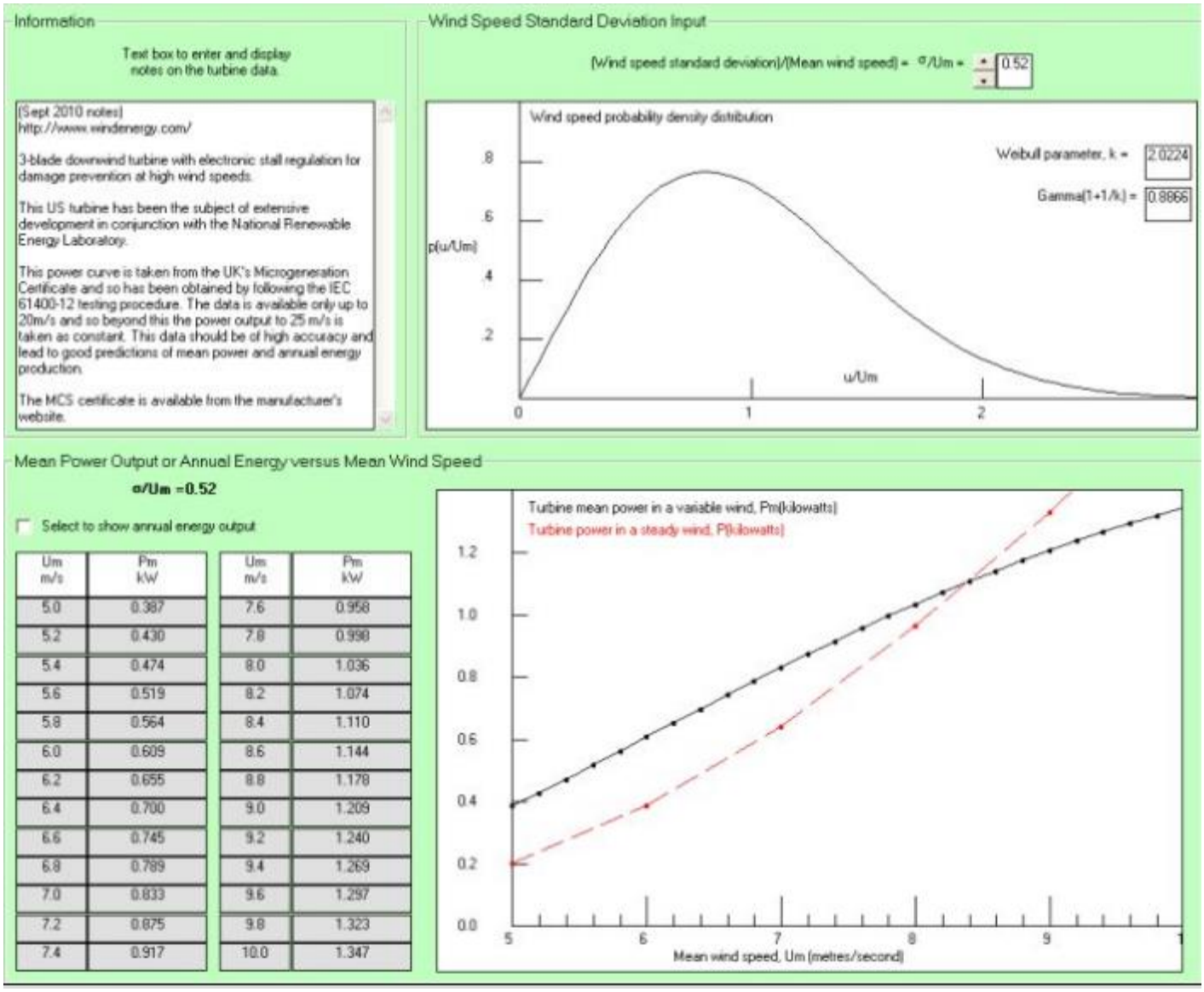
— власник землі (фермер) зацікавлений в середній і великій ВЕУ, який хоче, оцінити продуктивність і економічність її можливого встановлення;

— консультаційна фірма, що дає поради клієнтам з питань придбання, установки та застосування ВЕУ;

— організації, такі як школа, супермаркет або будь-яка інша компанія, з огляду на установку турбіни на їх території;

— спеціальні органи, які відповідальні за оцінку і моніторинг встановлення вітроенергетичних установок;

— студенти коледжів і університетів на курсах з відновлюваних джерел енергії.



Um m/s	Pm kW	Um m/s	Pm kW
5.0	0.387	7.6	0.958
5.2	0.430	7.8	0.998
5.4	0.474	8.0	1.036
5.6	0.519	8.2	1.074
5.8	0.564	8.4	1.110
6.0	0.609	8.6	1.144
6.2	0.655	8.8	1.178
6.4	0.700	9.0	1.209
6.6	0.745	9.2	1.240
6.8	0.789	9.4	1.269
7.0	0.833	9.6	1.297
7.2	0.875	9.8	1.323
7.4	0.917	10.0	1.347

Рисунок 2.2 — Результат роботи Wind Power

На середньому дисплеї потужності є графік, показаний як червона пунктирна лінія вихідної потужності ВЕУ при постійному вітру. Це показує, що при менших швидкостях середня вихідна потужність значно перевищує вихідну потужність при тій же стійкій швидкості вітру. Це є наслідком нелінійної поведінки кривої потужності та того факту, що середня потужність виграє від вітрових відхилень до частини кривої потужності. На відміну від цього, при високих середніх швидкостях

вітру, відхилення з високою швидкістю вітру переходять на сплющену область кривої потужності, що є результатом обмеження потужності електричних генераторів, з'єднаних з ротором. Середня потужність стає меншою, ніж стабільна потужність. Пункт перетину для більшості турбін становить близько 8 метрів в секунду.

Часто стверджується, що потужність ВЕУ змінюється, як куб швидкості вітру. Однак це неправильно і вводить в оману. Наявна потужність вітру, звичайно, змінюється, як куб швидкості вітру, але, як показано вище, турбіни не мають постійної ефективності в перетворенні цієї потенційної потужності в потужність валу.

2.2 Програмне забезпечення WAsP

Ця система є програмним забезпеченням для обчислення вітрових ресурсів, розміщення та розрахунку енерговитрат для ВЕУ та ВЕС. WAsP використовується для об'єктів, розташованих у всіх видах місцевості в усьому світі, і включає в себе моделі та інструменти для кожного етапу процесу від аналізу даних вітру до розрахунку енергетичного виходу для вітрової електростанції. Майже 5000 ліцензій WAsP було продано в близько 100 країнах [4].

Основні можливості програми WAsP (рисунок 2.3):

— розрахунок енергетичного виходу для окремих вітроенергетичних установок та вітроенергетичних станцій;

— розрахунок ефективності вітроелектростанцій;

— картографування вітрових ресурсів та відображення турбулентності по вибраних областях;

— розміщення вітрових турбін і вітрових електростанцій;

— розрахунок вітрових умов для оцінки МЕП-ділянки, наприклад середня швидкість вітру, нахил вітрового потоку, зсув вітру та турбулентність навколишнього середовища для одиночних вітрових турбін.

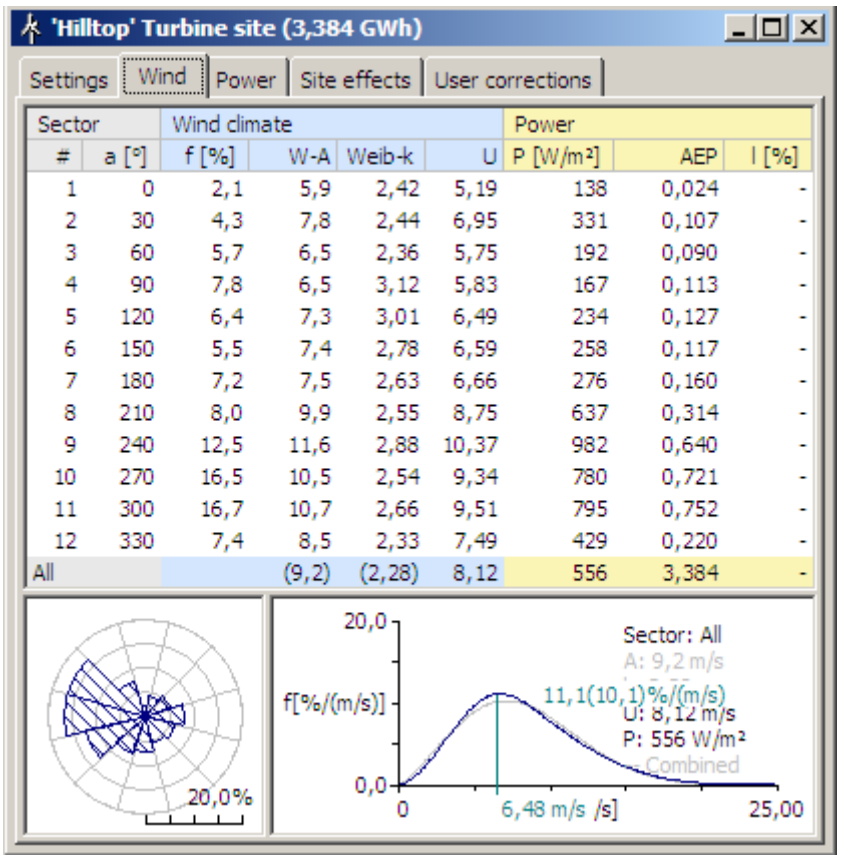


Рисунок 3.3 — Оцінка вітроенергетичного потенціалу, зроблена за допомогою WAsP

WAsP містить декілька фізичних моделей для опису вітрового клімату та вітрового потоку на різних територіях. Для горизонтальної та вертикальної екстраполяції WAsP використовує вбудовану лінійну модель, яка буде адекватно виконуватись від плоскої до помірно складної місцевості. Якщо місцевість дуже складна з багатьма крутими схилами, то WAsP також забезпечує легкий доступ до зовнішньої моделі.

WAsP поставляється в комплекті з безкоштовними утилітами для аналізу даних вітру, побудови та редагування цифрових карт, редагування кривої потужності та тяги, а також для оцінки місцевості IEC 61400-1 [5] та для вимірювання потужності IEC 61400-2-1 [5].

2.3 Висновки до розділу

У цьому розділі розглянуто дві програмні системи, які призначені для моніторингу та обчислення результатів роботи вітроенергетичних установок:

— Wind Power;

— WAsP.

Було визначено область застосування цих програмних систем, їх режими роботи та функціональні можливості.

6 ЗАСОБИ РОЗРОБКИ

При розробці програмного продукту важливим фактором є правильний вибір засобів програмної реалізації та технологій, які впливають на час розробки, якість та надійність продукту.

При створенні програми було обрано такі засоби:

— система керування базами даних Microsoft SQL Server 2017;

— середовище розробки Microsoft Visual Studio 2015;

— мову програмування C#;

— програмну технологію .NET Framework 4.5;

— технологію Windows Forms;

— програмний продукт ArcGIS;

Для розробки було використано ОС Windows 8.1, основними перевагами якої є:

— зручний інтерфейс;

— стабільність роботи системи;

— популярність використання.

Також операційна система Windows 8.1 має точки відновлення, тому в разі збою в роботі системи чи непередбачених обставин можливе повне відновлення документів та інших даних. Система є багатофункціональною, при встановленні має стандартний пакет драйверів [6].

Для розробки інтерфейсу користувача було обрано середовище Microsoft Visual Studio 2015 [7], використана база даних, що створена в MS SQL Server 2017 [8], для доступу до даних використано об’єктно-орієнтовану технологію ADO.NET [9]. Для розробки графічного середовища й інтерфейсу користувача використано технологію Windows Forms [10].

3.1 Система управління базами даних Microsoft SQL Server

Система управління базами даних Microsoft SQL Server є однією з найбільш популярних систем управління базами даних в світі. Дана СУБД підходить для найрізноманітніших проектів: від невеликих додатків до великих високонавантажених проектів.

СУБД SQL Server була створена компанією Microsoft. Перша версія вийшла в 1987 році. SQL Server довгий час був винятково системою управління базами даних для Windows, проте починаючи з версії 16 ця система доступна і на Linux. SQL Server характеризується такими особливостями як:

— продуктивність. SQL Server працює дуже швидко;

— надійність та безпека. SQL Server надає шифрування даних;

— простота. З даною СУБД відносно легко працювати і вести адміністрування.

Центральним аспектом в MS SQL Server, як і у будь-якій СУБД, є база даних. База даних являє собою сховище даних, організованих певним способом. Для організації баз даних MS SQL Server використовує реляційну модель. Ця модель баз даних була розроблена ще в 1970 році Едгаром Коддом. А на сьогоднішній день вона фактично є стандартом для організації баз даних.

Реляційна модель передбачає зберігання даних у вигляді таблиць, кожна з яких складається з рядків і стовпців. Кожен рядок зберігає окремий об’єкт, а в стовпчиках розміщуються атрибути цього об’єкта.

Для ідентифікації кожного рядка в рамках таблиці застосовується первинний ключ (primary key). В якості первинного ключа може виступати один або декілька стовпців. Використовуючи первинний ключ, ми можемо посилатися на певний рядок в таблиці. Відповідно два рядки не можуть мати один і той же первинний ключ.

Через ключі одна таблиця може бути зв’язана з іншою, тобто між двома таблицями можуть бути організовані зв’язки. А сама таблиця може бути представлена у вигляді відносин (“relation”).

Основний спосіб отримання даних з бази SQL Server — це запит. Запит описується за допомогою SQL. Запит декларативно вказує, що має бути отримано. Він обробляється процесором запиту, який з’ясовує послідовність кроків, які будуть

необхідні для отримання потрібних даних. Послідовність дій, необхідних для виконання запиту, називається планом запиту. Так може бути кілька способів обробки одного і того ж запиту. Наприклад, для запиту, який містить оператор вибору і оператор join, спочатку виконується join обох таблиць а потім вибір, або навпаки. В такому випадку, SQL Server вибирає план, який виконається швидше. Оптимізація запитів виконується безпосередньо в процесорі запиту.

Основна мова запитів для SQL Server — Transact-SQL [11], створений Microsoft разом із Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO для структурованої мови запитів (SQL) з розширеннями. T-SQL значно розширює можливості мови SQL і рекомендується до активного використання в найширшому спектрі завдань.

Залежно від завдання, яке виконує команда T-SQL, вона може належати до одного з наступних типів:

а) DDL (Data Definition Language / Мова визначення даних). До цього типу належать різні команди, які створюють базу даних, таблиці, індекси, збережені процедури і т.д. У загальних рисах визначають дані.

Зокрема, до цього типу ми можемо віднести наступні команди:

— CREATE: створює об’єкти бази даних (саму базу даних, таблиці, індекси і т.д.);

— ALTER: змінює об’єкти бази даних;

— DROP: видаляє об’єкти бази даних;

— TRUNCATE: видаляє всі дані з таблиць.

б) DML (Data Manipulation Language / Мова маніпуляції даними). До цього типу відносять команди на вибірку даних, їх оновлення, додавання, видалення — в загальному, це ті команди, за допомогою яких ми можемо керувати даними.

До цього типу належать такі команди:

— SELECT: витягує дані з БД;

— UPDATE: оновлює дані;

— INSERT: додає нові дані;

— DELETE: видаляє дані.

в) DCL (Data Control Language / Мова управління доступу до даних). До цього типу відносять команди, які керують правами щодо доступу до даних. Зокрема, це такі команди:

— GRANT: надає права для доступу до даних;

— REVOKE: відкликає права на доступ до даних.

Особливості T-SQL:

— не може використовувати псевдоніми виразів в операторах WHERE та HAVING;

— не може виконувати JOIN з використанням ключового слова USING();

— у разі наявності GROUP BY вимагає, щоб всі поля і вирази в секції SELECT або були явно присутні в GROUP BY, або були агрегатами;

— завжди вимагає явної вказівки приналежності поля до таблиці, якщо в двох і більше таблицях, що беруть участь у вибірці є однойменні поля;

— не підтримує функцію GROUP_CONCAT ();

— дозволяє не ставити крапку з комою в кінці оператора, для розділення операторів (і управління ходом виконання) використовує ключове слово GO;

— не підтримує конструкцію LIMIT, але підтримує TOP N і ROW_NUMBER ();

— крім UNION підтримує також EXCEPT і INTERSECT;

— підтримує FULL JOIN та CROSS APPLY.

3.2 Середовище розробки Visual Studio 2015

Середовище Microsoft Visual Studio — продукт розроблений компанією Microsoft, до якого входить інтегроване середовище розробки програмного забезпечення та ряд інших засобів розробки (рисунок 3.1).

Даний продукт дозволяє розробляти:

— додатки з графічним інтерфейсом з підтримкою технології Windows Forms;

— консольні додатки;

— веб-додатки;

— веб-служби;

— веб-сайти;

Ці додатки використовуються для всіх платформ, що підтримують Windows, .NET Framework, Silverlight та Windows Phone .NET Compact Framework.

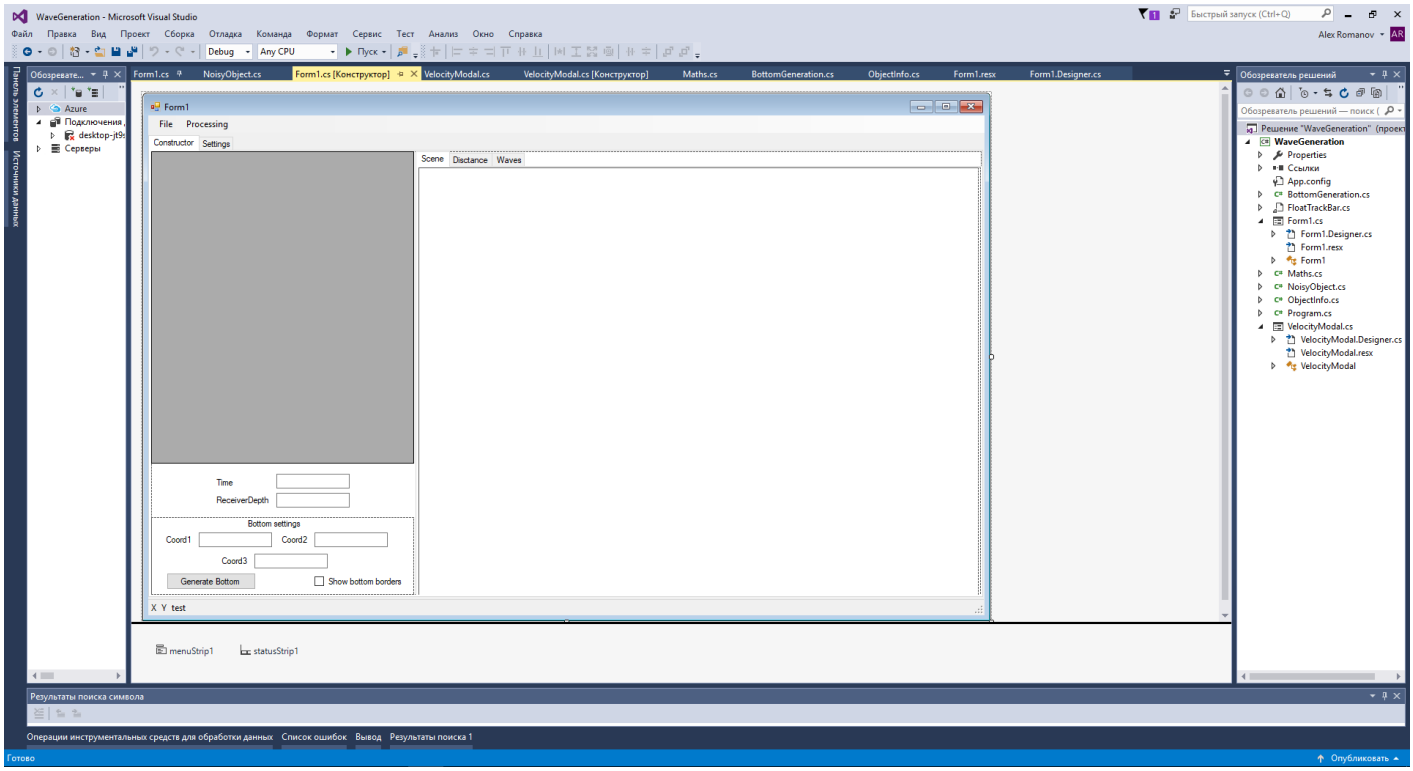


Рисунок 3.1 — Середовище розробки Microsoft Visual Studio 2015

Середовище Visual Studio містить інструменти які включають в себе редактор коду з можливостями рефакторінга коду та підтримкою технології IntelliSense. Також включає редактор форм для спрощеної розробки графічного інтерфейсу користувача, дизайнер схеми бази даних, дизайнер класів та веб-редактор. Вбудований відладчик може працювати як відладчик машинного рівня, так і відладчик рівня вихідного коду. У середовищі Visual Studio є можливість створення і підключення сторонніх додатків для розширення можливостей на кожному рівні, додавання нових наборів інструментів (наприклад, для візуального проектування та редагування), надає підтримку систем контролю версій вихідного коду та інструменти для деяких інших аспектів розробки програмних систем, наприклад такі як клієнт для роботи з Team Foundation Server —Team Explorer.

Середовище Visual Studio створено в архітектурі, яка підтримує можливість

використання доповнень (Add-Ins) — плагінів від сторонніх розробників, що дає змогу розширювати функції середовища розробки.

3.3 Мова програмування C#

Мова C# — об’єктно-орієнтована мова програмування яка має безпечну системою типізації для платформи .NET. Розроблена Скотом Вілтамутом, Пітером Гольде і Андерсом Гейлсбергом під егідою Microsoft Research (при фірмі Microsoft) [12].

За синтаксисом C# близький до Java та C++. Перейнявши багато особливостей від своїх попередників — мов Delphi, C++, Smalltalk і Модула — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне наслідування класів (на відміну від C++). Мова підтримує поліморфізм, перевантаження операторів, також має строгу статичну типізацію, вказівники на функції-члени класів, події, атрибути, винятки, коментарі у форматі XML.

Мова програмування C# розроблялась як мова прикладного рівня для CLR саме тому вона залежить, від можливостей самої CLR. Перш за все, це стосується системи типів C#. Наявність тих або інших виразних властивостей мови диктується тим, чи може конкретна мовна властивість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR до версії 2.0 значно збагатився і сам C# і надалі слід чекати подібної взаємодії. (Але ця послідовність буде порушена коли вийде C# 3.0, який є розширеннями мови, що не базується на розширення платформи .NET.) CLR пропонує C#, як і всім іншим .NET-орієнтованим мовам, багато функціональних можливостей, яких немає у «класичних» мовах програмування. Наприклад, в самому C# збірка сміття не реалізована, вона проводиться CLR для програм, які написані на C# аналогічно, як це робиться для програм на J#, VB.NET тощо.

Компілятором у C# є Microsoft Visual C#. Існують також інші компілятори C#, які часто включають реалізації Common Language Infrastructure і бібліотеки класів .NET:

— проект SharpDevelop який належить компанії icsharpcode, що використовується як альтернатива Visual Studio. Надає повну реалізацію Common Language Infrastructure. Останньою стабільною версією є IDE 4.4 (28 серпня 2013), а тестовою версією 5.0 (13 лютого 2014). За зовнішнім виглядом IDE схожа на Microsoft Visual C#, що дозволяє перейти від одного середовища до іншого без зайвих труднощів[13];

— проект Microsoft Rotor (який зараз називається Shared Source Common Language Infrastructure, ліцензований тільки для дослідницького та навчального використання) надає реалізації CLR runtime та компілятор C# та підмножину бібліотек фреймворка Common Language Infrastructure, відповідно до специфікації ECMA (з підтримкою тільки Windows XP, до C# 2.0 [14];

— проект DotGNU так само надає відкритий компілятор C#, а також схожу до повної реалізацію Common Language Infrastructure, включаючи частину деяких залишених особистих бібліотек класів Microsoft .NET до .NET 2.0 (які є включеними у стандартний Microsoft .NET Framework, проте не є документованими або не включені до специфікації ECMA) та потрібні бібліотеки фреймворка відповідно до специфікації ECMA [15];

— проект Mono, започаткований компанією Xamarin, і куплений згодом її наступником Novell, надає відкритий компілятор C#, відкриту та повну реалізацію Common Language Infrastructure, також включає всі необхідні бібліотеки фреймворка згідно із специфікацією ECMA, а також близьку до повної реалізацію особистих бібліотек класів Microsoft .NET до версії .NET 2.0, але не специфічних бібліотек .NET 3.0 та .NET 3.5, як для Mono 2.0 [16];

— DotNetAnywhere Micro Framework CLR направлений на вбудовані системи та підтримує практично всі специфікації C# 2.0.

Розглянемо деякі особливості мови програмування C#:

— першою особливістю мови C# є портативність. C# створювалась як мова програмування на прикладному рівні для Common Language Runtime через це вона в першу чергу залежить від функціоналу самої CLR [17]. В першу чергу це має відношення до типізації в C#. Наявність тих чи інших особливостей мови залежить від того, чи може та чи інша мовна особливість бути трансльована у конкретні конструкції CLR. Так, з розвитком CLR від версії 1.1 також значно збільшилися можливості мови програмування C#; такої ж взаємодії можна було б чекати і в майбутньому. (Але ця послідовність буде порушена, коли вийде C# 3.0, що є доповненням мови, яке не опирається на розширення платформи .NET.) CLR дає мові C#, так як і всім іншим .NET-орієнтованим мовам програмування, багато різних можливостей, яких позбавлені «класичні» мови програмування;

— другою особливістю вважають типи даних у мові C#. Розглянемо головні типи даних у таблиці 3.1;

Таблиця 3.1 — Основні типи даних мови програмування C#

Type	Represents	Range	Default Value
Bool	Boolean value	True or False	False
Byte	8-bit unsigned integer	0 to 255	0
Char	16-bit Unicode character	U +0000 to U +ffff	'\0'
Decimal	128-bit precise decimal values with 28-29 significant digits	(-7.9 x 1028 to 7.9 x 1028) / 100 to 28	0.0M
Double	64-bit double-precision floating point type	(+/-)5.0 x 10-324 to (+/-)1.7 x 10308	0.0D

Float	32-bit single-precision floating point type	-3.4 x 1038 to + 3.4 x 1038	0.0F
Int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
Long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
Sbyte	8-bit signed integer type	-128 to 127	0
Short	16-bit signed integer type	-32,768 to 32,767	0
Uint	32-bit unsigned integer type	0 to 4,294,967,295	0

Продовження таблиці 3.1

Type	Represents	Range	Default Value
Ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
Ushort	16-bit unsigned integer type	to 65,535	0

— третьою особливістю в мові C# є метапрограмування. Метапрограмування через використання атрибутів у C# є особливістю цієї мови. Деякі з цих атрибутів повторюють функціональні особливості директив препроцесора, які орієнтовані на платформу VisualC ++ та GCC [18].

3.4 Технологія .NET

Технологія Microsoft .NET — програмна технологія, заснована фірмою Microsoft як платформа для створення як веб-застосунків, так і звичайних програм. Багато принципів та ідей запозичені з технології Java. Сумісність служб, що написані різними мовами є однією з головних ідей .NET. Хоч ця можливість вказується Microsoft як перевага .NET, але у платформі Java є така ж можливість.

Програма, написана мовою C#, виконується в середовищі .NET Framework (рисунок 3.2) — інтегрованому компоненті для Windows, який містить віртуальну систему виконання (CLR-середовище), уніфікований набір бібліотек класів для всіх мов платформи .NET — BCL і бібліотеку класів FCL з розширеними бібліотеками: ASP.NET [19], ADO.NET, Windows Forms, WPF [20].

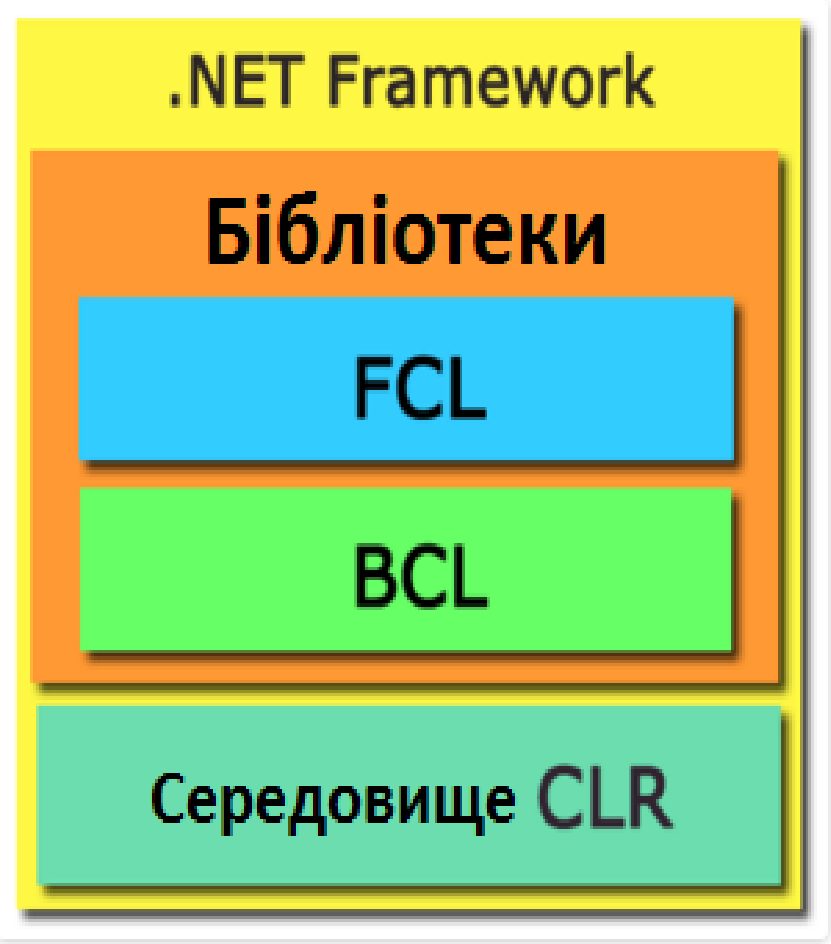


Рисунок 3.2 — Компоненти платформи .NET Framework

Кожна бібліотека в .NET має свою власну версію, що допомагає уникнути появу можливих конфліктів між різними версіями збірок.

.NET — крос-платформенна технологія, на даний момент існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) та варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell).

Захист авторських прав відноситься до створення середовищ виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET випускаються фірмами для різних мов у вільному доступі.

.NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки.

Середовище CLR (рисунок 3.3) є комерційною реалізацією інфраструктури CLI (common language infrastructure), міжнародного стандарту, основи середовищ виконання та розробки з тісною взаємодією мов і бібліотек.

Вихідний код, написаний мовою C #, компілюється в проміжну мову (IL) відповідно до специфікації CLI. Код IL і ресурси, такі як растрові зображення і рядки, зберігаються на диску в виконуваному файлі-збірці з розширенням EXE або DLL у більшості випадків.

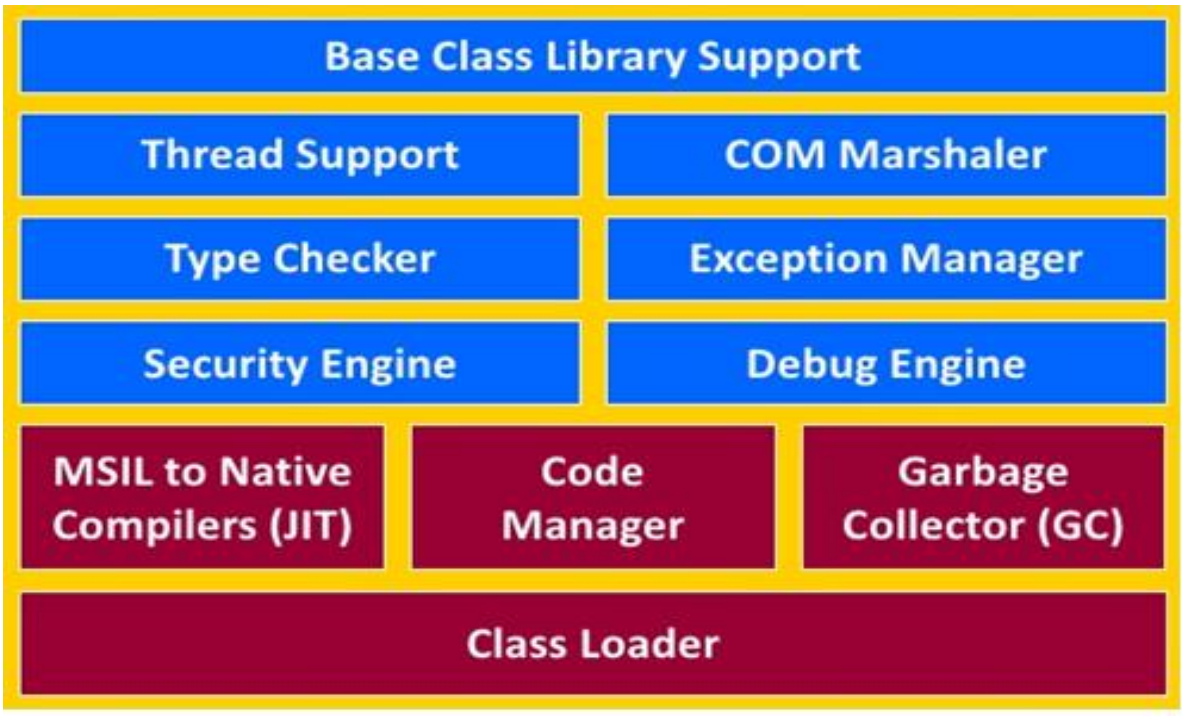


Рисунок 3.3 —Архітектура середовища CLR

При виконанні C#-програми збірка завантажується в збірку CLR згідно з відомостями в маніфесті. Далі, якщо вимоги безпеки дотримано, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, які стосуються автоматичного збору сміття, обробки виключень і керування ресурсами.

Також один з перших Java компіляторів був розроблений фірмою Microsoft (зараз в Java використовується HotSpot — набагато досконаліша багаторівнева компіляція). Від якості того чи іншого компілятора часто залежить питання швидкодії, тому сучасна технологія динамічної компіляції дає можливість отримати аналогічного рівня швидкодії з традиційними “статичними” компіляторами (наприклад, C++) .

3.5 Технологія Windows Forms

Технологія Windows Forms — інтерфейс програмування додатків (API), що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код — класи, що реалізують API для Windows Forms, не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C#, C ++, так і на VB.Net, J# та інших.

З одного боку, Windows Forms розглядається як заміна більш старої і складної бібліотеці MFC, спочатку написаної на мові C++. З іншого боку, WF не пропонує парадигму, порівнянну з MVC. Для виправлення цієї ситуації і реалізації даної функціональності в WF існують сторонні бібліотеки.

Усередині .NET Framework, Windows Forms реалізується в рамках простору імен System.Windows.Forms.

Додаток Windows Forms є подієво-орієнтованим, що підтримується Microsoft .NET Framework. На відміну від пакетних програм, велика частина часу витрачається на очікування від користувача будь-яких дій, як, наприклад, введення тексту в текстове поле або кліка мишкою по кнопці.

Для подальшого розвитку корпорація Майкрософт домоглася успіху у використанні WinForms з XAML за допомогою таких фреймворків, як WPF та UWP [21]. Проте перетягування компонентів графічного інтерфейсу способом, подібним до WinForms, все ще забезпечується за допомогою XAML, замінюючи кореневий елемент XAML сторінки або вікна за допомогою інтерфейсу користувача Canvas. При виконанні цієї зміни користувач може побудувати вікно аналогічним чином як і у WinForms шляхом безпосереднього перетягування компонентів за допомогою графічного інтерфейсу Visual Studio. На рисунку 3.4 показано приклад Windows Forms.

Хоча XAML забезпечувала перетягування та зворотну сумісність місць розташування за допомогою контрольної панелі Canvas, слід зазначити, що елементи керування XAML подібні до WinForm Controls і не сумісні з один в один. Вони виконують подібні функції та мають подібний зовнішній вигляд, але властивості та методи досить різні, щоб вимагати перенаправлення з одного API на інший.

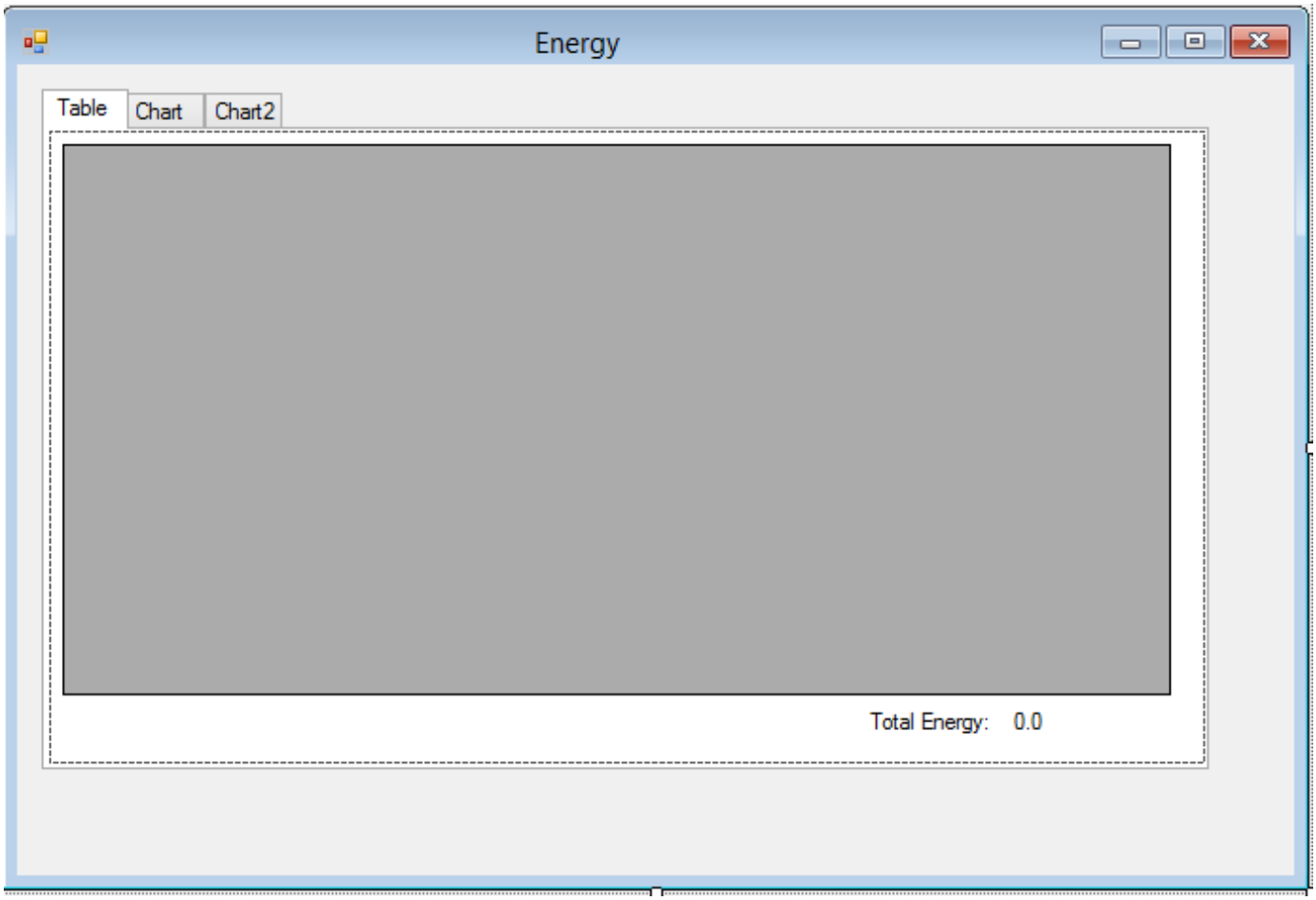


Рисунок 3.4 — Приклад Windows Forms

3.6 Технологія ArcObjects

Програмні продукти компанії ESRI (США), найстарішого у світі виробника програмних засобів ГІС (фірма заснована в 1969 році).

Програмні продукти Arcgis порівняно з іншими програмами роботи з географічною інформацією мають ряд переваг:

— строга топологічність даних;

— контроль за цілісністю і топологічністю даних;

— розвинутий апарат роботи з системою координат і географічними проекціями;

— наявність розвинутого математичного апарату обробки просторових даних.

Основними її недоліками є деякі труднощі при освоєнні редактора, а також заплутана і не завжди зрозуміла ліцензійна політика і велика вартість ліцензій.

Програма ArcObjects це середовище розробки сімейства ArcGIS додатків [22]. За допомогою Visual Basic для додатків, C Sharp (мова програмування) або Java (мова програмування) SDK для ArcGIS, що дозволяє розробникам розширювати ці додатки.

Програма ArcObjects є бібліотекою COM-компонентів, що створюють фундамент платформи ArcGIS від ESRI. ArcObjects написана в основному на мові програмування C++. Всі ArcGIS для настільних додатків засновані на ArcObjects. Оскільки ArcGIS повністю побудований на вершині ArcObjects, ви можете використовувати COM-послуги та можливості, щоб повністю налаштувати і розширити платформу ArcGIS-meaning, щоб розширення моделі ArcObjects даних можна зробити легко і практично з усіма COM-сумісними мовами програмування (наприклад, Visual Basic, C#, Visual Basic.NET, Java і Python). COM надає компоненти для повторного використання на бінарному рівні. Іншими словами, розробники не вимагають доступу до вихідного коду ArcObjects для того, щоб розширити платформу ArcGIS. З цієї причини при ArcObjects програміст може використовувати будь-який тип всередині системи ArcObjects, не знаючи деталей реалізації типу. Розробник тільки повинен знати, що тип може зробити.

Оскільки ArcObjects заснований на стандарті COM, ви можете легко працювати з ним в поєднанні з іншими COM-об’єктами і додатками (багато програмних додатків на базі Windows, такі як Microsoft Office засновані на стандарті COM). Як уже згадувалося раніше, платформа ArcGIS була побудована з використанням типів ArcObjects (такі як класи, інтерфейси і перерахування). У світі ArcObjects, класи використовують інтерфейси для організації властивостей і методів. Простіше кажучи, класи всередині ArcObjects використовують тільки COM-інтерфейси, щоб виставити їх відкриті члени і спілкуються один з одним.

При роботі з класом ArcObjects COM, ви ніколи не працювати з властивостями і методами класу; швидше, ви завжди отримати доступ до його властивостей і методів за допомогою одного зі своїх реалізованих інтерфейсів. Як приклад, для екземпляра об’єкта ви можете використовувати тільки один інтерфейс. Однак, після

конкретизації, ви можете запросити будь-який інший інтерфейс, який реалізується за допомогою цього об’єкта. Цей процес іноді називається інтерфейс Query (QI). Класи в ArcObjects часто мають багато інтерфейсів. Багато з компонентів ArcObjects, які складають ArcGIS використовуються у всіх трьох продуктах ArcGIS. Об’єкти в широких категоріях базових послуг, доступ до даних, аналіз карти і карт презентацій містяться у всіх трьох продуктах. Ці чотири категорії містять велику частину функціональних можливостей ГІС та надаються розробникам і користувачам ArcGIS.

3.7 Вимоги до системи

Рекомендованими версіями операційних систем для нормальної роботи додатку є Windows 7, Windows 8 та Windows 10. Для запуску програми на вашому персональному комп’ютері має бути встановлена версія .NET Framework не нижче 4.5.

3.8 Висновки до розділу

У цьому розділі було розглянуто засоби, що були застосовані для розробки проекту. Оскільки розроблена система призначена для роботи у ОС Microsoft Windows, було обрано відповідну мову програмування та середовище розробки. Технології, які використовуються, були обрані за принципом зручності у використанні, відкритості вихідних кодів та актуальності в наш час. Дані технології в сукупності дають можливість побудувати якісний та надійний продукт, який захищений від патентних посягань з боку розробників, бо всі ці технології покриті ліцензіями, які виключають таку можливість і надають доступ до вихідних кодів даних проектів.

7 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У цьому розділі розглянуто загальний опис розробленої системи, та особливості програмної реалізації.

Розроблена система включає в себе такі логічні модулі:

— керуючий модуль;

— модуль управління базою даних;

— модуль управління картою;

— база даних;

— ADO.NET;

— модуль обробки результатів;

— модуль виводу результатів.

На рисунку 4.1 наведена схема структури системи, на якій розташовані всі програмні модулі.

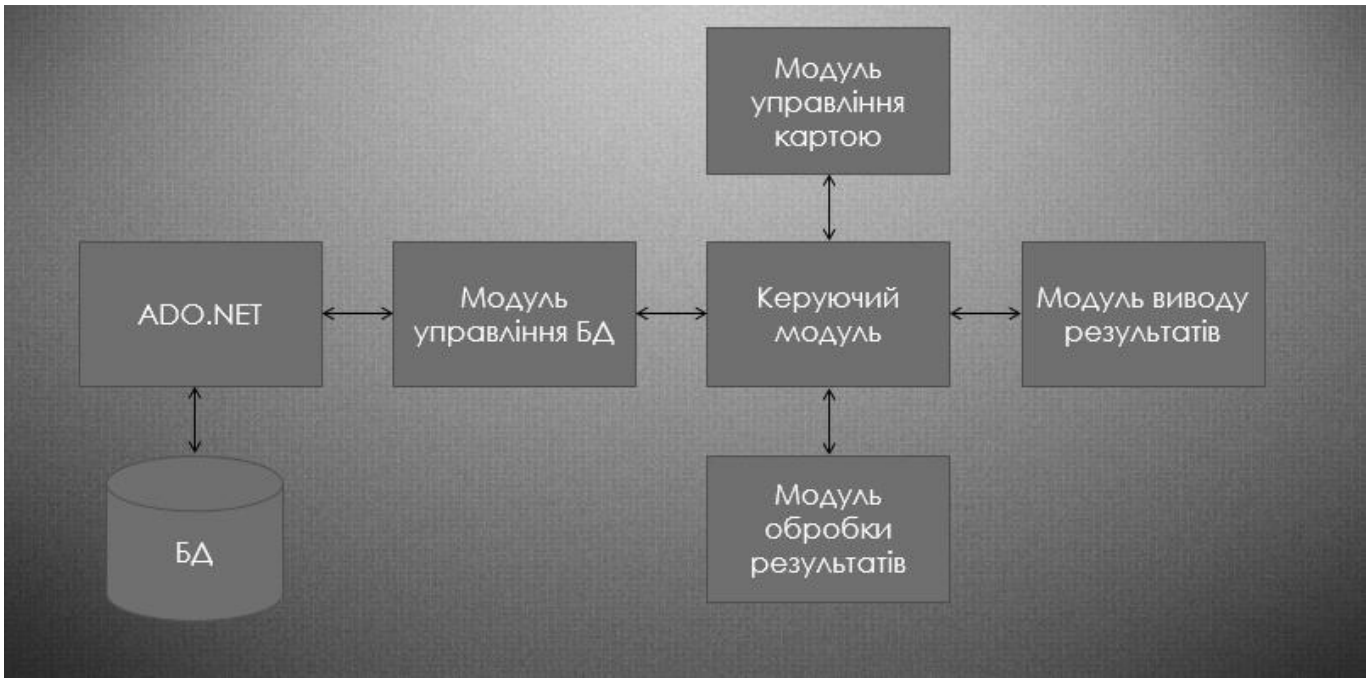


Рисунок 4.1 — Схема структури системи

4.1 Концептуальна модель бази даних

База даних системи складається з п’ятьох взаємопов’язаних таблиць реляційної бази даних, які створюють єдиний інформаційний простір для зберігання та отримання доступу до даних.

Головною таблицею є “Показник”, яка містить у собі інформацію про метеорологічні умови, в першу чергу про вітрову активність.

Таблиця “Тип показника” містить основну інформацію про кожен конкретний показник, його назву та одиниці вимірювання.

Таблиця “Регіон” містить інформацію про конкретну територію із вказанням коду території та її назви.

Таблиця “Вітряк” містить інформацію про модель та виробника.

Таблиця “Технічна характеристика вітряка” містить інформацію про технічні можливості вітряка, а саме про його потужність в залежності від різних вітрових умов, тощо.

Концептуальна модель бази даних приведена на рисунку 4.2 та 4.3.

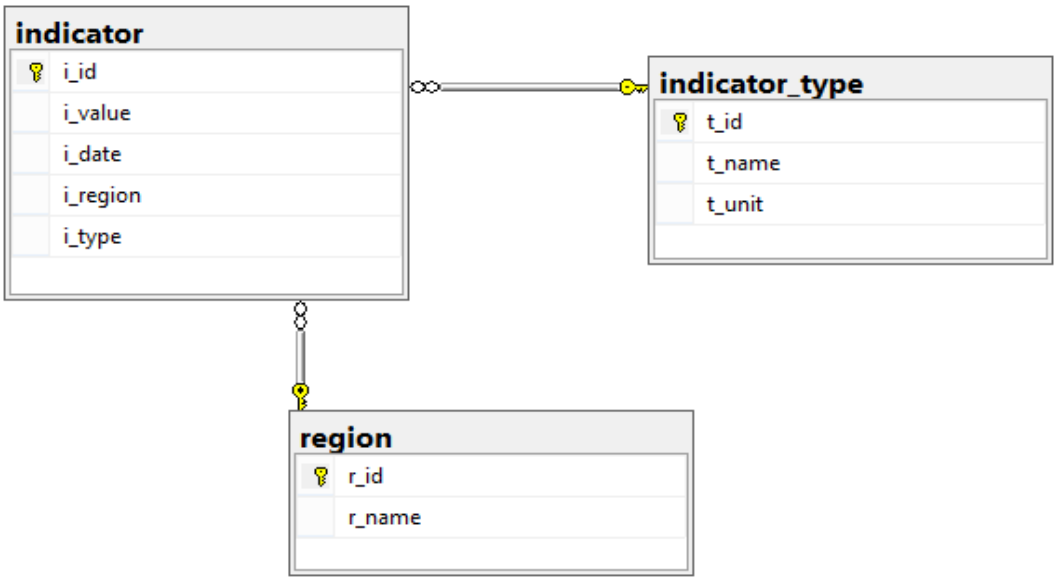


Рисунок 4.2 — Концептуальна модель БД показників

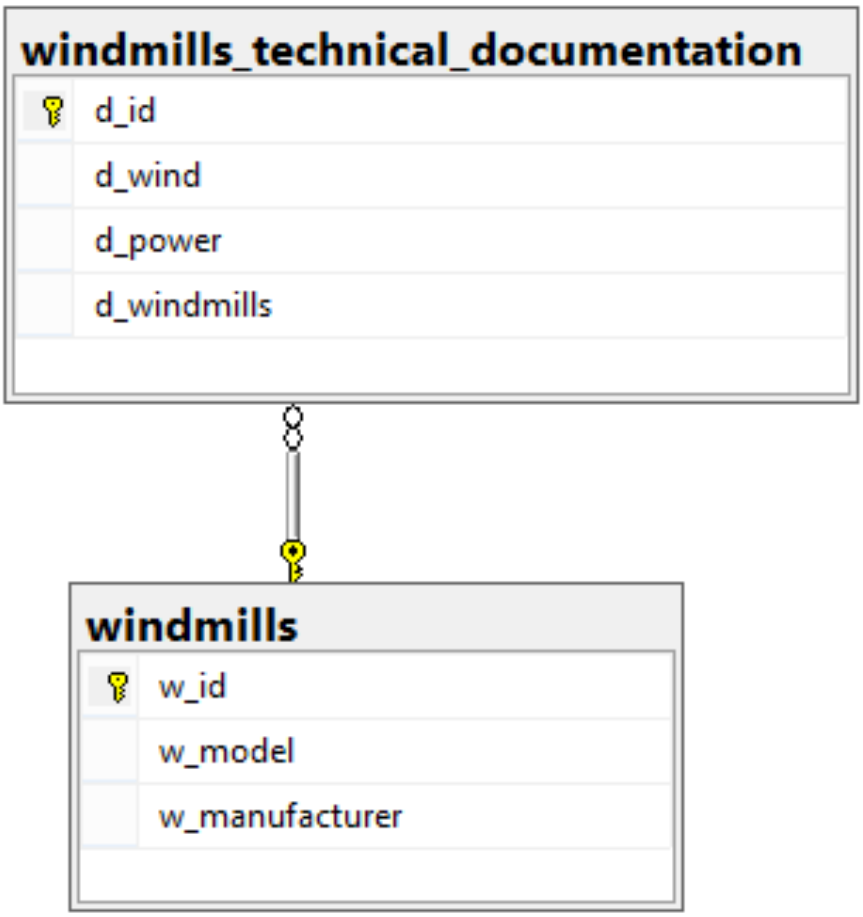


Рисунок 4.3 — Концептуальна модель БД вітряків

4.2 Опис таблиць бази даних

Для доступу до даних із бази для кожної таблиці створюється звичайний клас на мові C# з публічними властивостям, який є об’єктним відображенням таблиці в базі даних.

DbContext забезпечує об’єктно-орієнтований інтерфейс для доступу і маніпулювання даними, що зберігаються в базах даних. Клас DbSet відповідає таблиці в базі даних, поле моделі являє собою значення окремого стовпця рядка.

Також використання об’єктно-реляційної проекції дозволяє зручно отримувати данні з таблиць, які мають зв’язок “багато-до-багатьох” та “один-до-багатьох”.

Розглянемо більш детально структури кожної із таблиць бази даних.

Детальна інформація про їх структури (ім’я, тип і розмір поля, опис поля) приведена у таблицях 4.1 — 4.5.

Таблиця 4.1. Структура таблиці “Показник”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
[i_id]	INT	Первинний ключ
[i_value]	NVARCHAR(255)	Значення показника
[i_date]	DATETIME	Дата вимірювання
[i_region]	NVARCHAR(255)	Регіон вимірювання показника
[i_type]	INT	Зовнішній ключ на таблицю тип показника

Таблиця 4.2. Структура таблиці “Тип показника”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
[t_id]	INT	Первинний ключ
[t_name]	NVARCHAR(255)	Назва типу показника
[t_unit]	NVARCHAR(255)	Одиниці вимірювання

Таблиця 4.3. Структура таблиці “Регіон”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
[r_id]	NVARCHAR(255)	Первинний ключ, код території
[r_name]	NVARCHAR(255)	Назва території

Таблиця 4.4. Структура таблиці “Вітряк”

Ім’я поля	**Тип і розмір поля**	**Опис поля**

[w_id]	INT	Первинний ключ
[w_model]	NVARCHAR(255)	Модель вітряка
[w_manufacturer]	NVARCHAR(255)	Виробник вітряка

Таблиця 4.5. Структура таблиці “ Технічна документація вітряка ”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
[d_id]	INT	Первинний ключ
[d_wind]	INT	Швидкість вітру
[d_power]	DECIMAL(10,2)	Потужність вітряка
[d_windmills]	INT	Зовнішній ключ на таблицю вітряк

4.4 Опис алгоритму програми

Покроково розглянемо задачу обчислення генерації енергії вітроенергетичною установкою (рисунок 4.4).

Перш за все необхідно визначити початкові дані, які знадобляться нам у подальших розрахунках. Для виконання цієї задачі необхідно створити зручний користувацький інтерфейс, за допомогою якого користувач програмною системою зможе задати початкові параметри оцінки.

Після того як користувач обрав всі необхідні параметри, програма витягує дані з бази метеорологічних показників та дані про вітроенергетичну установку та її характеристику.

З однієї сторони ми отримуємо швидкість вітру та згідно вказаного користувачем періоду вимірювання обчислюємо тривалість режиму.

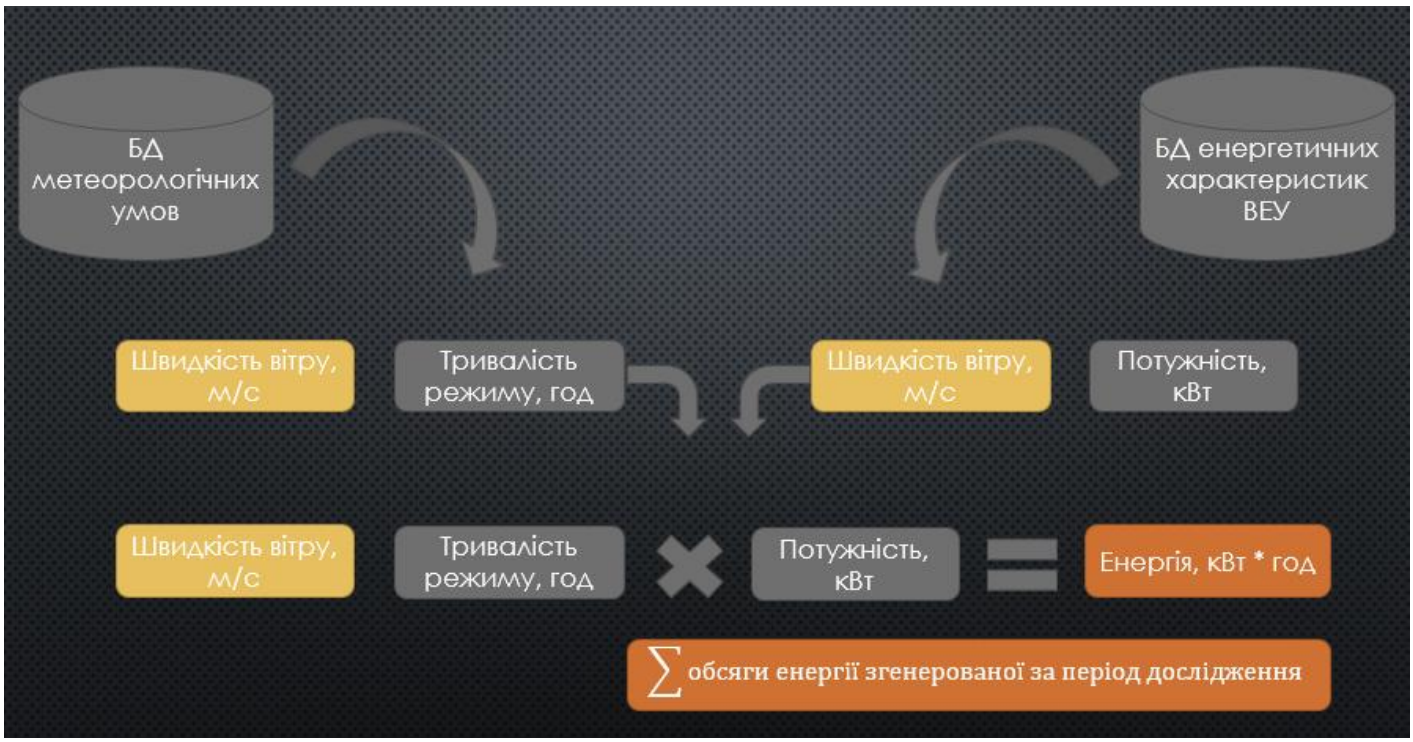


Рисунок 4.4 — Алгоритм роботи програми

З іншої сторони ми маємо потужність вітроенергетичної установки, яка залежить від швидкості вітру. Перемноживши тривалість вітрового режиму та потужність ВЕУ для цієї швидкості вітру ми отримуємо кількість виробленої енергії за конкретну одиницю швидкості вітру.

Просумувавши ці результати ми отримаємо сумарні обсяги згенерованої енергії за період дослідження.

4.4 Висновки до розділу

У цьому розділі описано загальний вигляд системи, модулі з яких вона складається. Також було розглянуто структуру бази даних системи, детально розглянуто таблиці та їх описи. Було показано загальний алгоритм роботи програми.

8 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для забезпечення нормальної та стабільної роботи системи, що дозволяє розрахувати обсяги генерування енергії за визначений проміжок часу на основі метеорологічних умов (вітрова активність) та бази енергетичних характеристик треба дотримуватися основних вимог при інсталяції та рекомендацій щодо її використання.

5.1 Системні вимоги

Для встановлення розробленої програмної системи персональний комп'ютер повинен мати процесор Intel® Pentium® / Core ™ / Celeron ® чи AMD™ Athlon™ / Turion ™ з тактовою чистотою не нижче 2.2 GHz, та 2 GB ОЗУ. Рекомендованими версіями операційних систем для нормальної роботи додатку є Windows 7, Windows 8 та Windows 10. Для запуску програми на вашому персональному комп'ютері має бути встановлена версія .NET Framework не нижче 4.5. На жорсткому диску повинно бути не менше 50 Мб вільного місця.

5.2 Інструкція з використання програмного продукту

Інтерфейс програми виконано за допомогою Windows Forms, він включає в себе безліч елементів управління для вводу або вибору даних, вкладок, панелей та графіків для відображення інформації.

При запуску програми користувачу відображається карта України в програмі ArcGIS (рисунок 5.1). Користувач повинен натиснути лівою клавішею миші на одну з територій, де він хоче розрахувати кількість виробленої енергії.

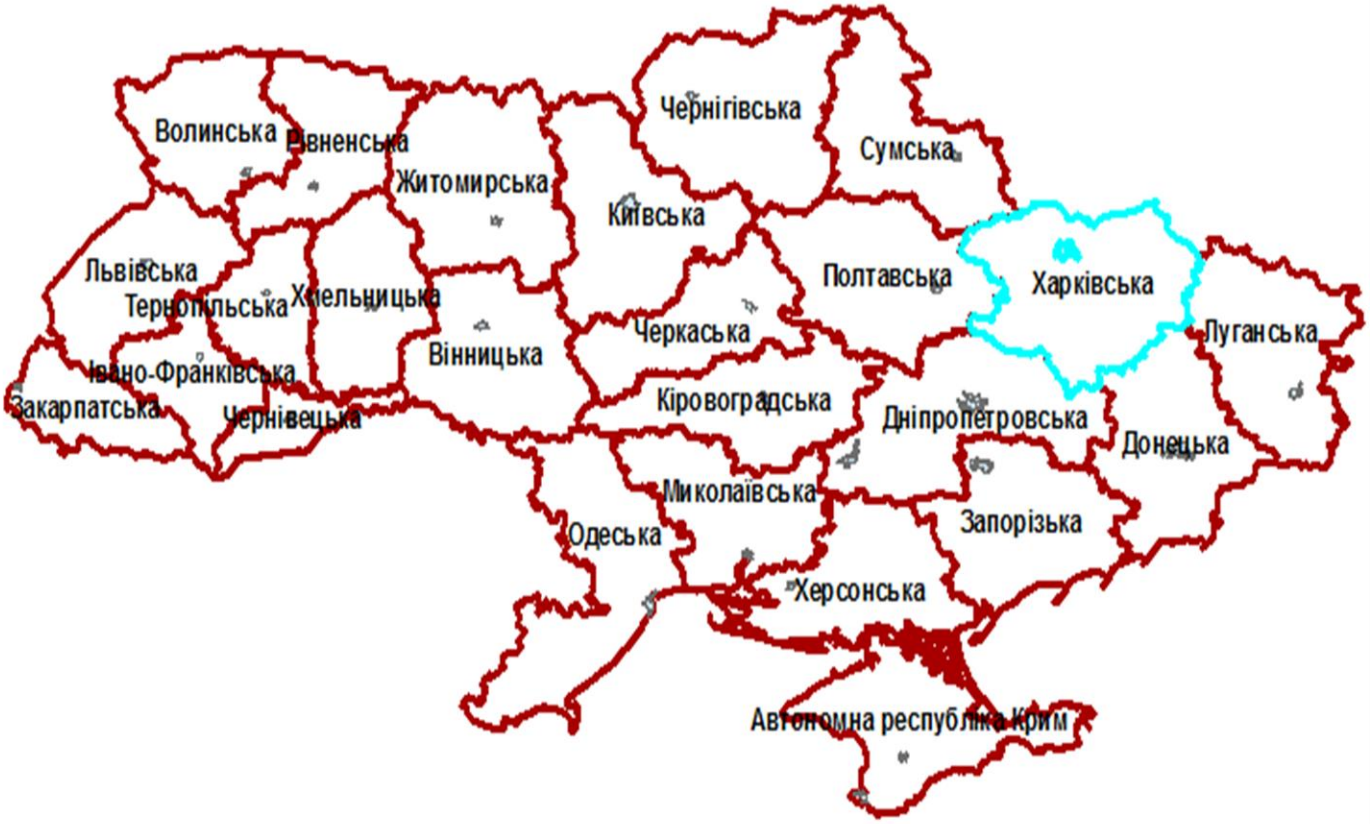


Рисунок 5.1. — Загальний вигляд головної сцени, з відміченою територією.

При повторному натисненні лівої кнопки буде встановлена цільова точка, а також повторне натиснення спричинить виклик модального вікна (рисунок 5.2), де користувачу пропонується вибрати діапазон (з якої по яку дату) буде проводитись подальша оцінка. За допомогою DateTimePicker можна обрати відповідні дати.

В процесі оцінки можна вибирати будь-який діапазон — тиждень, місяць, рік. Та на підставі даних метеорологічних показників проводити обчислення обсягів генерування енергії за обраний період на обраній території.

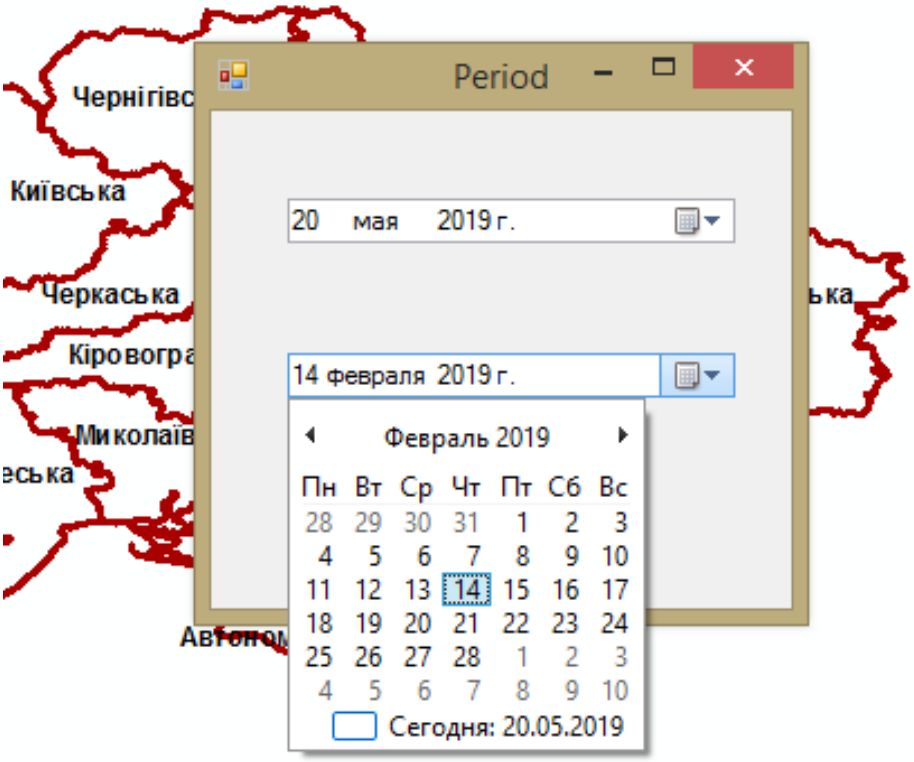


Рисунок 5.2 — Форма вибору діапазону дослідження

Після того, як користувач вибрав відповідні дати, буде викликано відповідний модуль, який зробить запит до бази даних для отримання необхідних даних за вказаний період. Після цього відбудеться розрахунок на основі якого користувачу буде надана інформація про розподіл вітрового потенціалу за швидкостями (рисунок 5.3) у вигляді графіку та у табличному вигляді. Також можна переглянути інформацію по технічній характеристиці ВЕУ, а саме криву потужності в залежності від швидкості вітру (рисунок 5.4).

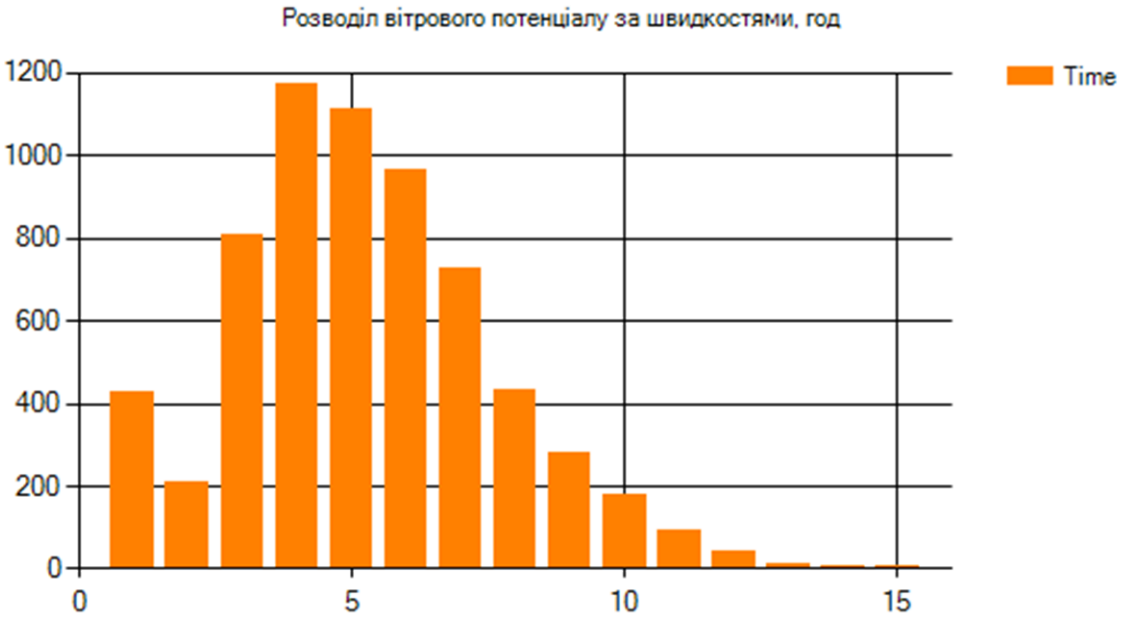


Рисунок 5.3 — Графік розподілу вітрового потенціалу за швидкостями

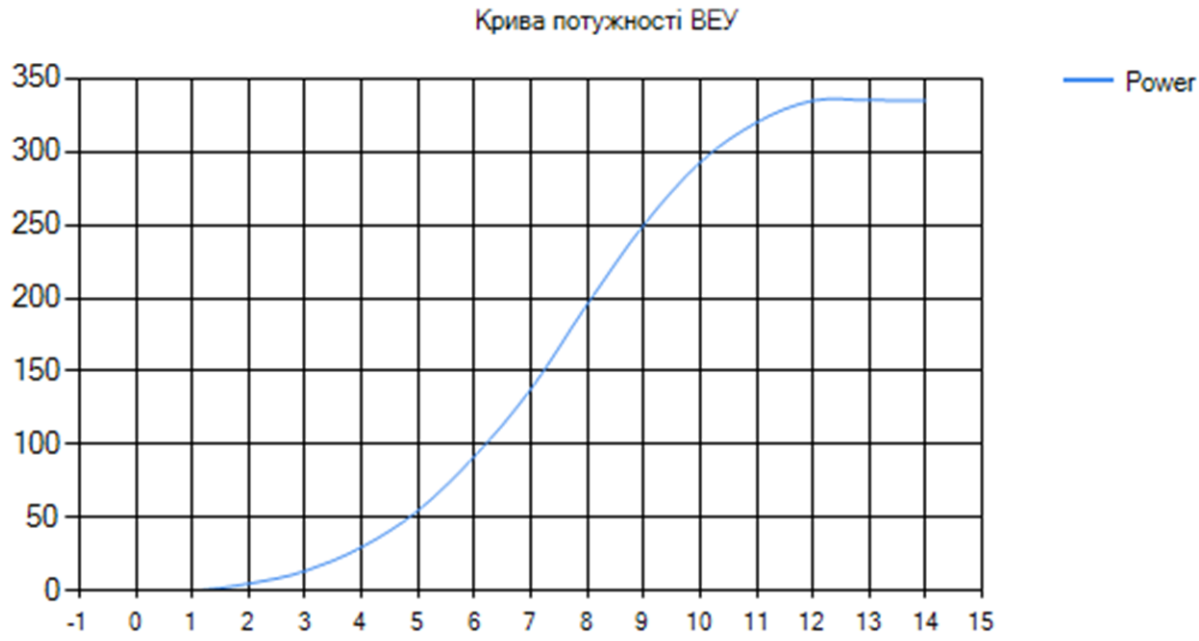


Рисунок 5.4 — Крива потужності ВЕУ

Також користувач може переглянути таблицю де наведена вся необхідна інформація пов’язана із розрахунком виробленої енергії. (рисунок 5.5).

	Швидкість вітру, м/с	Сумарна тривалість, год	Потужність ВЕУ, кВт	Енергія вироблена ВЕУ, кВт*год
	0	57	0	0
	1	18	0	0
	2	66	5	330
	3	98	13,7	1342,6
	4	117	30	3510
	5	87	55	4785
	6	73	92	6716
	7	77	138	10626
	8	41	196	8036
	9	33	250	8250
	10	27	292,8	7905,6

Total Energy: 58981,2

Рисунок 5.5 — Таблиця генерації електричної енергії

Під таблицею знаходяться сумарні обсяги енергії згенерованої за період дослідження. Користувач має змогу вибирати інші території та порівнювати скільки буде згенеровано сумарної енергії.

5.3 Системні вимоги

У цьому розділі наведено інформацію про системні вимоги для нормального встановлення та користування програмним продуктом. Було надано інструкцію з використання, що дозволить користувачу швидше зрозуміти загальні принципи роботи програмної системи. Також задачу надання користувачу інформації про

взаємодію з програмою виконують скріншоти, наведені для спрощення розуміння виконаних у розділі кроків.

ВИСНОВКИ

У ході виконання роботи було створено програмний продукт, що дозволяє розрахувати обсяги генерування енергії за визначений проміжок часу на основі метеорологічних умов (вітрова активність) та бази енергетичних характеристик.

Вхідні дані система отримує через елементи керування головного інтерфейсу, за допомогою яких користувач вводить необхідні для розрахунку дані.

Під час виконання роботи було виконано наступні задачі:

1. Створено гнучку структуру бази даних метеорологічних показників та додано дані для проведення оцінки.
2. Створено базу даних вітроенергетичних установок і додано їх технічні характеристики.
3. Створено інтерфейс для зручної взаємодії користувача з програмною системою, який дозволяє вводити початкові дані для оцінки. Також за допомогою створеного інтерфейсу користувач має можливість ввести усі дані, що стосуються вітроенергетичної установки. Для кращого сприйняття отриманих даних, після розрахунку створюються графіки, що відображають найважливішу інформацію у зручному для користувача вигляді.
4. Розраховано обсяги генерування енергії за період дослідження з урахуванням обраної території.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Wind Power [Електронний ресурс] — Режим доступу: http://www.wind-power-program.com.

2. Wind Turbine Components. Danish Wind Industry Association. [Електронний ресурс] — Режим доступу: https://winddenmark.dk/en/tour/wtrb/comp/index.htm.

3. Raciti Castelli, Marco; Englaro, Alessandro; Benini, Ernesto (2011). The Darrieus wind turbine: Proposal for a new performance prediction model based on CFD. [Електронний ресурс] — Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0360544.

4. Документація WAsP [Електронний ресурс] — Режим доступу: https://www.wasp.dk/wasp.

5. Woebbeking, Mike. IEC TS 61400-22 [Електронний ресурс] — Режим доступу: http://www.gl-group.com/pdf/IEC_TS_61400-22_Woeb.

6. Документація Microsoft Windows 8.1 [Електронний ресурс] — Режим доступу: https://www.microsoft.com/uk-ua/software/windows8ISO.

7. Документація Microsoft Visual Studio [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/visualstudio.

8. Документація MS SQL Server 2017 [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/sql/?view=sql-server-2017.

9. Документація технології ADO.NET [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet.

10. Документація Windows Forms [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/dotnet/framework/winforms.

11. Документація діалекту Transact-SQL [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-2017.

12. Документація мови програмування C# [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide.

13. Документація SharpDevelop Project [Електронний ресурс] — Режим доступу: http://www.icsharpcode.net/OpenSource/SD/Default.aspx.

14. Документація Microsoft Rotor Project [Електронний ресурс] — Режим доступу: https://blogs.msdn.microsoft.com/jasonz/2006/03/23/rotor-sscli-2-0-ships.

15. Документація DotGNU Project [Електронний ресурс] — Режим доступу: https://www.gnu.org/software/dotgnu/free-software.html.

16. Документація Mono Project [Електронний ресурс] — Режим доступу: https://www.mono-project.com/docs.

17. Документація Common Language Runtime [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/dotnet/standard/clr.

18. Harald Sondergaard. Course on Program Analysis and Transformation [Електронний ресурс] — Режим доступу: http://archive.handbook.unimelb.edu.au/view/2013/comp90053.

19. Документація ASP.NET [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/aspnet.

20. Документація WPF [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/dotnet/framework/wpf.

21. Документація UWP [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/windows/uwp.

22. Документація ArcObjects [Електронний ресурс] — Режим доступу: http://help.arcgis.com/en/sdk/10.0/arcobjects_net/ao_home.html.

ДОДАТОК 1

Створення БД для аналізу території з метою розміщення вітроенергетичних станцій

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_ТМ52 19Б 81-1	Литка_С. С._ТМ52.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_ТМ52 19Б 12-1	/Controllers/*.cs /Models/*.cs /Services/*.cs	Модулі серверної частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_ТМ52 19Б 12-2	/Form/*.cs	Модуль клієнтської частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_ТМ52 19Б 13-2	Опис.docx	Опис модуля інтерфейсу клієнтської частини програми

ДОДАТОК 2

Створення БД для аналізу території з метою розміщення вітроенергетичних станцій

Текст програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52_19Б 12-1

Аркушів 9

Київ – 2019

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using ESRI.ArcGIS.Desktop.AddIns;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.ArcMapUI;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindPowerApp
{
   public class WindPow : ESRI.ArcGIS.Desktop.AddIns.Tool
   {
      public WindPow ()
      {
      }
      protected override void OnMouseDown(MouseEventArgs arg)
      {

         //if (arg.Button == System.Windows.Forms.MouseButtons.Left)
          // check for left mouse button if you like
         // convert point from 'screen' to 'map' coordinates
         IPoint MouseMapPoint = (ArcMap.Document.FocusMap as
IActiveView).ScreenDisplay.DisplayTransformation.ToMapPoint(arg.X, arg.Y);

         // clear selection first
         ArcMap.Document.FocusMap.ClearSelection();

         // Select using the shape (point) to
         // select the feature(s) - false to select any intersecting, true to select just the first
         ArcMap.Document.FocusMap.SelectByShape(MouseMapPoint,
(ArcMap.Application as IMxApplication).SelectionEnvironment, false);

         // get the (now) selected features
         IEnumFeature EnumFeatures =
(IEnumFeature)ArcMap.Document.FocusMap.FeatureSelection;
         IEnumFeatureSetup efs = EnumFeatures as IEnumFeatureSetup;
         efs.AllFields = true;
         IFeature ThisFeature = EnumFeatures.Next();
         int fldKoatuu = ThisFeature.Fields.FindField("CODEOBJ");
```

```
do
        {
            // get the geometry
            IGeometry ThisGeom = ThisFeature.ShapeCopy; // IMPORTANT! USE
SHAPECOPY not SHAPE
                                                // do something different for each geometry type
            if (ThisGeom.GeometryType == esriGeometryType.esriGeometryPoint)
            {
               // something for points (note: does not include multipoint)
            }
            else if (ThisGeom.GeometryType == esriGeometryType.esriGeometryPolygon)
            {
               // something for polygons
               if (ThisFeature.Class.AliasName == "BigCity1")
               {
                  string _koatuu = ThisFeature.Value[fldKoatuu];
                  var period = new Period(_koatuu);
                  period.Show();
               }
            }
            else if (ThisGeom.GeometryType == esriGeometryType.esriGeometryPolyline)
            {
               // something for polylines (note: polylines are different to lines)
            }
            else
            {
               // unrecognized geometry type
            }

            SqlConnection mySqlCon = new SqlConnection();
            ThisFeature = EnumFeatures.Next();

        } while (ThisFeature != null);
        ArcMap.Document.ActiveView.Refresh();
        base.OnMouseDown(arg);
      }
      protected override void OnUpdate()
      {
         Enabled = ArcMap.Application != null;
      }
   }

}
```

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindPowerApp
{
   public static class Data
   {
      private static string connectionString = @"Data
Source=SERHIY\SQLEXPRESS;Initial Catalog=wind_power_station;Integrated
Security=True";

      private static List<indicator> Indicators = new List<indicator>();

      private static List<region> Regions = new List<region>();

      private static List<windmills> Windmills = new List<windmills>();

      private static List<windmills_technical_documentation> WindmillsDoc = new
List<windmills_technical_documentation>();

      public static List<DataModel> Calculate(DateTime dateFrom, DateTime dateTo,
string region)
      {
         var res = new List<DataModel>();
         var col_1_2 = Indicators.Where(x => (int)x.type == 1 && (string)x.region ==
region &&
            (DateTime)x.date >= dateFrom && (DateTime)x.date <= dateTo)
            .GroupBy(x => x.value)
            .Select(x => new { Speed = int.Parse(x.Key.ToString()),
                  Time = (double) x.Count() / 2 })
            .OrderBy(x => x.Speed).ToList();

         var col_3 = WindmillsDoc.Take(col_1_2.Count)
                  . Where(x => (int)x.windills == 1)
                  .OrderBy(x => x.wind).Select(x => double.Parse(x.power.ToString()))
                  .ToList();
```

```
for (int i = 0; i < col_1_2.Count; i++)
            {
                res.Add(new DataModel()
                {
                    Speed = col_1_2[i].Speed,
                    Time = col_1_2[i].Time,
                    Power = col_3[i],
                    Energy = col_1_2[i].Time * col_3[i],
                });
            }
            return res;
        }

        public static void GetData()
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string cmdTextIndicators = "SELECT * FROM [dbo].[indicator]";
                SqlCommand cmdIndicators = new SqlCommand(cmdTextIndicators,
connection);
                SqlDataReader readerIndicators = cmdIndicators.ExecuteReader();
                if (readerIndicators.HasRows)
                {
                    while (readerIndicators.Read())
                    {
                        Indicators.Add(new indicator()
                        {
                            id = readerIndicators.GetValue(0),
                            value = Math.Abs(int.Parse(readerIndicators.GetValue(1).ToString())),
                            date = readerIndicators.GetValue(2),
                            region = readerIndicators.GetValue(3),
                            type = readerIndicators.GetValue(4),
                        });
                    }
                }
                readerIndicators.Close();
                string cmdTextRegions = "SELECT * FROM [dbo].[region]";
                SqlCommand cmdRegion = new SqlCommand(cmdTextRegions, connection);
                SqlDataReader readerRegions = cmdRegion.ExecuteReader();
                if (readerRegions.HasRows)
                {
                    while (readerRegions.Read())
                    {
                        Regions.Add(new region()
```

```
                {
                    id = readerRegions.GetValue(0),
                    name = readerRegions.GetValue(1),
                });
            }
        }
        readerRegions.Close();
        string cmdTextWindmills = "SELECT * FROM [dbo].[windmills]";
        SqlCommand cmdWindmills = new SqlCommand(cmdTextWindmills,
connection);
        SqlDataReader readerWindmills = cmdWindmills.ExecuteReader();
        if (readerWindmills.HasRows)
        {
            while (readerWindmills.Read())
            {
                Windmills.Add(new windmills()
                {
                    id = readerWindmills.GetValue(0),
                    model = readerWindmills.GetValue(1),
                    manufacturer = readerWindmills.GetValue(2),
                });
            }
        }
        readerWindmills.Close();
        string cmdTextWindmillsDoc = "SELECT * FROM
[dbo].[windmills_technical_documentation]";
        SqlCommand cmdWindmillsDoc = new SqlCommand(cmdTextWindmillsDoc,
connection);
        SqlDataReader readerWindmillsDoc = cmdWindmillsDoc.ExecuteReader();
        if (readerWindmillsDoc.HasRows)
        {
            while (readerWindmillsDoc.Read())
            {
                WindmillsDoc.Add(new windmills_technical_documentation()
                {
                    id = readerWindmillsDoc.GetValue(0),
                    wind = readerWindmillsDoc.GetValue(1),
                    power = readerWindmillsDoc.GetValue(2),
                    windills = readerWindmillsDoc.GetValue(3),
                });
            }
        }
        readerWindmills.Close();
    }
```

```
public class DataModel
{
    public int Speed { get; set; }

    public double Time { get; set; }

    public double Power { get; set; }

    public double Energy { get; set; }
}

class indicator
{
    public object id { get; set; }

    public object value { get; set; }

    public object date { get; set; }

    public object region { get; set; }

    public object type { get; set; }
}

class region
{
    public object id { get; set; }
    public object name { get; set; }
}

class windmills
{
    public object id { get; set; }
    public object model { get; set; }
    public object manufacturer { get; set; }
}

class windmills_technical_documentation
{
    public object id { get; set; }
    public object wind { get; set; }
    public object power { get; set; }
    public object windills { get; set; }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindPowerApp
{
   public partial class Period : Form
   {
      public string Code { get; set; }
      public Period()
      {
         InitializeComponent();
      }
      public Period(string code)
      {
         InitializeComponent();
         Data.GetData();
         Code = code;
      }

      private void SetPeriod_Click(object sender, EventArgs e)
      {
         new Energy(Data.Calculate(DateFrom.Value, DateTo.Value,
Code)).ShowDialog();
         Close();
      }
   }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
```

```
namespace WindPowerApp
{
    public partial class Energy : Form
    {
        public Energy()
        {
            InitializeComponent();
        }

        public Energy(List<DataModel> data)
        {
            InitializeComponent();
            DataGrid.DataSource = data;
            DataGrid.Columns[0].HeaderText = "Швидкість вітру, м/с";
            DataGrid.Columns[1].HeaderText = "Сумарна тривалість, год";
            DataGrid.Columns[2].HeaderText = "Потужність ВЕУ, кВт";
            DataGrid.Columns[3].HeaderText = "Енергія вироблена ВЕУ, кВт*год";

            TotalEnergy.Text = data.Sum(x => x.Energy).ToString();

            ChartEnergy.Titles.Add("Розводіл вітрового потенціалу за швидкостями, год");
            ChartEnergy.Palette = ChartColorPalette.EarthTones;
            ChartEnergy.Series.Clear();
            Series series = ChartEnergy.Series.Add("Time");
            for (int i = 0; i < data.Count; i++)
            {
                series.Points.Add(data[i].Time);
            }

            ChartEnergy2.Titles.Add("Крива потужності ВЕУ");
            ChartEnergy2.Series.Clear();
            Series series2 = this.ChartEnergy2.Series.Add("Power");
            series2.ChartType = SeriesChartType.Spline;
            for (int i = 0; i < data.Count; i++)
            {
                series2.Points.AddXY(data[i].Speed, data[i].Power);

            }
        }
    }
}
```

ДОДАТОК 3

Створення БД для аналізу території з метою розміщення вітроенергетичних станцій

Опис програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52_19Б 13-2

Аркушів 8

Київ – 2019

АНОТАЦІЯ

Додаток надає можливість розрахувати обсяги генерування енергії за визначений проміжок часу на основі метеорологічних умов та енергетичних характеристик вітроенергетичної установки.

Розроблене програмне забезпечення дозволяє отримати візуальне представлення результатів у вигляді графіків та таблиць.

Для розробки програмного забезпечення було використано середовище розробки Microsoft Visual Studio 2015, мову програмування C#, програмну технологію .NET Framework 4.5, програмний продукт ArcGIS, а також технологію Windows Forms для створення інтерфейсу користувача. Системою керування базами даних було обрано MS SQL Server 2017.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до теми дипломної роботи, програма має назву «Сервіс для обчислення кількості згенерованої енергії вітроенергетичною установкою».

Для встановлення розробленої програмної системи персональний комп’ютер повинен мати процесор Intel® Pentium® / Core ™ / Celeron ® чи AMD™ Athlon™ / Turion ™ з тактовою чистотою не нижче 2.2 GHz, та 2 GB ОЗУ. Рекомендованими версіями операційних систем для нормальної роботи додатку є Windows 7, Windows 8 та Windows 10. Для запуску програми на вашому персональному комп’ютері має бути встановлена версія .NET Framework не нижче 4.5. На жорсткому диску повинно бути не менше 50 Мб вільного місця..

Система була написана мовою програмування C#, з використанням фреймворку .NET Framework 4.5, продукту ArcGIS, а також технології Windows Forms для створення інтерфейсу користувача.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб покликаний вирішити задачу аналізу території з метою розміщення вітроенергетичних станцій. Це було реалізовано за допомогою кількох функцій системи, а саме:

— вибору території для оцінки;

— вводу періоду на протязі якого буде здійснюватись оцінка;

— розрахунку обсягів генерування енергії за визначений проміжок часу;

— відображення графічної візуалізації результатів.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальний принцип роботи додатку такий:

1) користувач обирає територію;
2) вводить період дослідження;
3) метод, що викликається в обробнику виконує запит до БД;
4) на основі отриманих даних проводиться обчислення кількості згенерованої енергії;
5) результати виконання виводяться на сторінку.

В першу чергу завантажується карта в ArcGis. Користувач обирає територію та вводить всі необхідні дані.

Після натискання на кнопку “Розрахувати”, що розміщена під формою, викликається метод Calculate, в який передаються всі початкові параметри. Він виконує запит до бд та отримує всі ці дані. Потім починає проводити розрахунки за написаним алгоритмом. Після успішного завершення він повертає результат у вигляді структури як відображається як таблиця.

Після цього користувач може подивитись візуалізацію результатів, натиснувши на конпку відповідного графіка.

Після закінчення роботи з цими даними, користувач може очистити всі введені початкові дані і ввести нові.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для організації доступу до програмного продукту потрібно мати комп’ютер або ноутбук.

Для роботи з програмою кінцевим користувачам потрібно, щоб на комп’ютері була встановлена версія .NET Framework не нижче 4.5. На жорсткому диску повинно бути не менше 50 Мб вільного місця.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— метеорологічні показники;

— технічні характеристики вітроенергетичної установки;

— територія для аналізу;

— період протягом якого буде здійснюватись аналіз.

Вихідними даними є:

— розраховані обсяги генерування енергії;

— візуалізація результатів у вигляді графіків.